②

AL-TR-1992-0114

AD-A262 796

ARMSTRONG LABORATORY

MODELING STUDENT KNOWLEDGE WITH SELF-ORGANIZING FEATURE MAPS

Steven A. Harp
Tariq Samad
Michael Villano

Honeywell Sensor and System Development Center
3660 Technology Drive
Minneapolis, MN 55418

HUMAN RESOURCES DIRECTORATE
TECHNICAL TRAINING RESEARCH DIVISION
7909 Lindbergh Drive
Brooks Air Force Base, TX 78235-5352

February 1993

Final Technical Report for Period February 1991 - April 1992

AIR FORCE MATERIEL COMMAND
BROOKS AIR FORCE BASE, TEXAS

NOTICES

[illegible]

JIMMY L. FLEMING
Project Scientist

[illegible]

JAMES W. PARLETT, Major, USAF
Chief, Intelligent Training Branch

[illegible]

RODGER D. BALLENTINE, Colonel, USAF
Chief, Technical Training Research Division

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** March 1993	**3. REPORT TYPE AND DATES COVERED** Final – February 1991 – April 1992

4. TITLE AND SUBTITLE Modeling Student Knowledge with Self-Organizing Feature Maps	**5. FUNDING NUMBERS** C – F33615-91-C-0002; PE – 62205F; PR – 1121; TA – 09; WU – 81
6. AUTHOR(S) Steven A. Harp; Tariq Samad; Michael Villano	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Honeywell Sensor and System Development Center; 3660 Technology Drive; Minneapolis, MN 55418	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES) Armstrong Laboratory; Human Resources Directorate; Technical Training Research Division; 7909 Lindbergh Drive; Brooks Air Force Base, TX 78235-5352	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** AL-TR-1992-0114

1. SUPPLEMENTARY NOTES

Armstrong Laboratory Technical Monitor: Jimmy L. Fleming, (210) 536-2034

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (Maximum 200 words)

This report describes a novel application of neural networks to model the behavior of students in the context of an intelligent tutoring system. Self-organizing feature maps are used to capture the possible states of student knowledge from an existing test database. The trained network implements a universal student knowledge model that is compatible with recently developed Knowledge Space Theory approaches to student assessment and computer aided instruction. The student model can be applied to rapidly assess the knowledge of any given student, and chart a path from lower to higher states of expertise. We illustrate the concept on an aircraft fuel management domain, demonstrating its noise-tolerance and insensitivity to feature map parameter values. An approach to determining the correct feature map size is also described.

14. SUBJECT TERMS Artificial neural networks; Intelligent computer aided instruction; Intelligent tutoring systems; Student modeling	**15. NUMBER OF PAGES** 32
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std Z39-18
298-102

CONTENTS

FIGURES

CONTENTS (Continued)

MODELING STUDENT KNOWLEDGE WITH SELF-ORGANIZING FEATURE MAPS

INTRODUCTION

Software for education has become increasingly sophisticated in recent years. The terms "Intelligent Computer Assisted Instruction" (ICAI) and "Intelligent Tutoring System" (ITS) are used to describe new educational programs that incorporate the techniques of artificial intelligence (Sleeman & Brown, 1982; Wenger, 1987; Wood & Holt, 1990). A distinction of these programs over previous instructional systems is that they attempt to explicitly represent knowledge about the domain of instruction and about the students they are teaching. This concept enables them to reason about the most effective ways to communicate the concepts of the domain to a given student.

A critical component of an intelligent tutoring system is the *student model.* In the student model, a theory of student behavior, the ITS predicts what a student will do in a given context. This model may include a wide range of information: what the student knows or doesn't know, any misconceptions ("bugs") the student may harbor, the student's "cognitive style," degree of forgetfulness, receptivity to advice, etc.

Current approaches to student modeling are either ad hoc or make strong assumptions regarding the student and problem domains, and require substantial expertise to implement. This paper presents an alternative approach to overcome these limitations. A resulting model, referred to as a model of the universe of student knowledge (MUSK), captures the capabilities of students of different mastery levels and also indicates learning paths from lower to higher levels. An advantage over many other approaches to student modeling is that MUSK can have an empirical genesis: MUSKs can be derived from automatic data analysis rather than the labor of human experts.

There is a connection between our MUSKs and models that have been developed in the psychological literature, particularly the *knowledge space theory* of Falmagne and Doignon (Doignon & Falmagne, 1985; Villano & Bloom, 1992). In knowledge space theory, an area of expertise can be quantized into *items,* which may be test questions or equivalence classes of questions. The *knowledge state* of a student is defined as the set of items the student can correctly answer. Generally, a small fraction of the full power set of imaginable states are realizable in the population of students since, in many domains of knowledge, being able to handle some items implies being able to handle certain others. For example, in arithmetic, knowing how to multiply fractions implies knowing how to multiply integers. Thus, any state containing the capability to multiply fractions will also contain the capability to multiply integers. The collection of knowledge states actually realizable by students forms a *knowledge structure.* The "structure" is apparent in the lattice of subset relations between the member knowledge states. Falmagne and associates have explored still stronger mathematical models (c.f. Falmagne, 1989). Discovering a general knowledge structure in test data is the problem we have addressed.

Figures 1 through 3 outline our approach. The first step in the development of a universal student knowledge model is the collection of a problem response database (Fig. 1). The database includes the responses of a population of students to a number of standard problems. Both private and governmental institutions already possess such databases. For example, the New York State Board of Regents has collected the results of subject-specific comprehensive examinations given to hundreds of thousands of high school students each year. Similarly, national testing services collect the results of standardized achievement and aptitude tests.

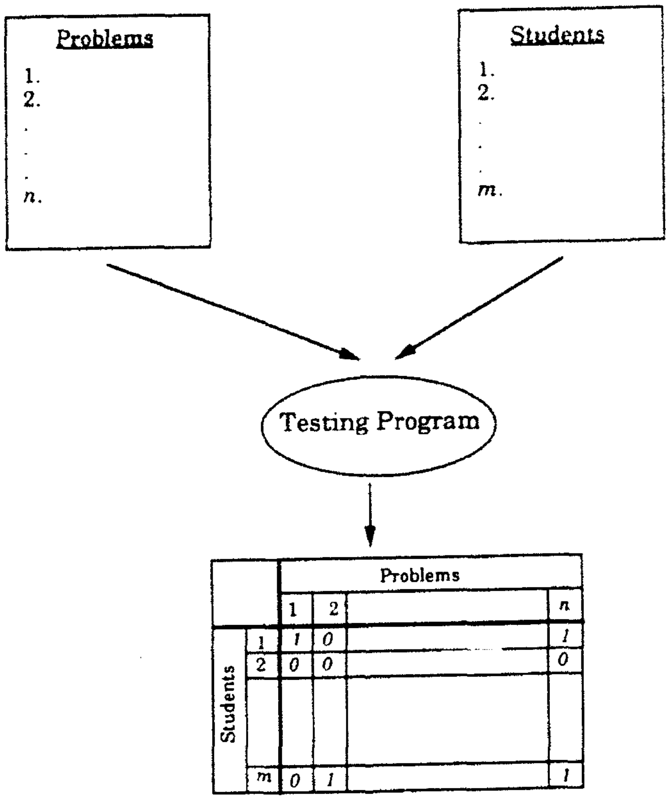

Figure 1
Development of a problem response database.
Each of *m* students is posed *n* problems.

The problem response database is used to develop a neural-network-based universal student knowledge model (Fig. 2). The student model identifies the knowledge states of the domain and organizes them into a graph that shows their interrelationships. The graph may or may not conform to representations commonly used in the psychometric or assessment community. The common psychometric assumption is rather strong; namely that the states are uniformly graded on a single dimension of skill (a Guttman scale; Lord, 1980; Weiss, 1983). Constraints adopted in knowledge space theory include the assumption that a null state and a state with complete knowledge exist. A weaker sort of model constrains states to be closed under union and to be well-graded. While such assumptions are not built into our student modeling approach, a MUSK model can readily be converted to a knowledge structure or a knowledge space representation. To the extent that approaches to the application of student models are based on such assumptions, however, such a conversion may be desirable.

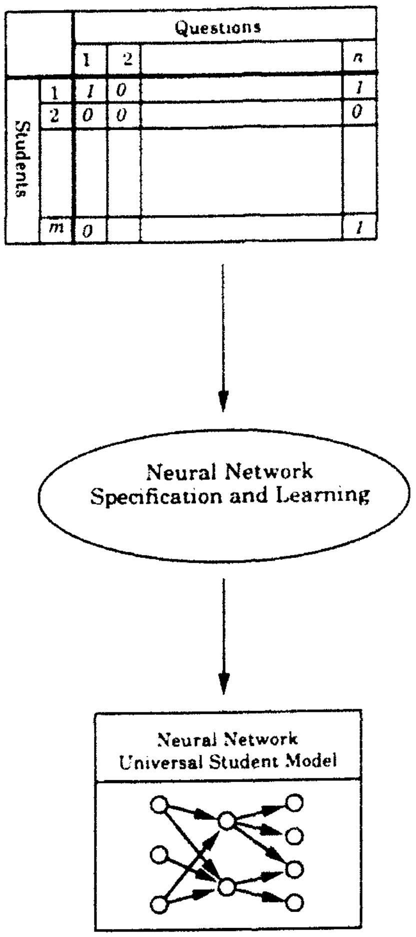
Questions
Students
1
2
n
m
Neural Network Specification and Learning
Neural Network Universal Student Model

Figure 2
Developing a neural network implementation of a MUSK model from a response database.

Once the student model is developed, it provides a concise representation of the distribution of student capabilities. We foresee the student model as an integral component of an ITS used to support an instructional module (Fig. 3).

The focus of this paper is on the development of a student model using neural network learning methods from problem response data. Later, we briefly discuss some promising ways for an ITS to benefit from such a student model. A more complete discussion of this latter topic appears in Villano & Bloom, 1992.

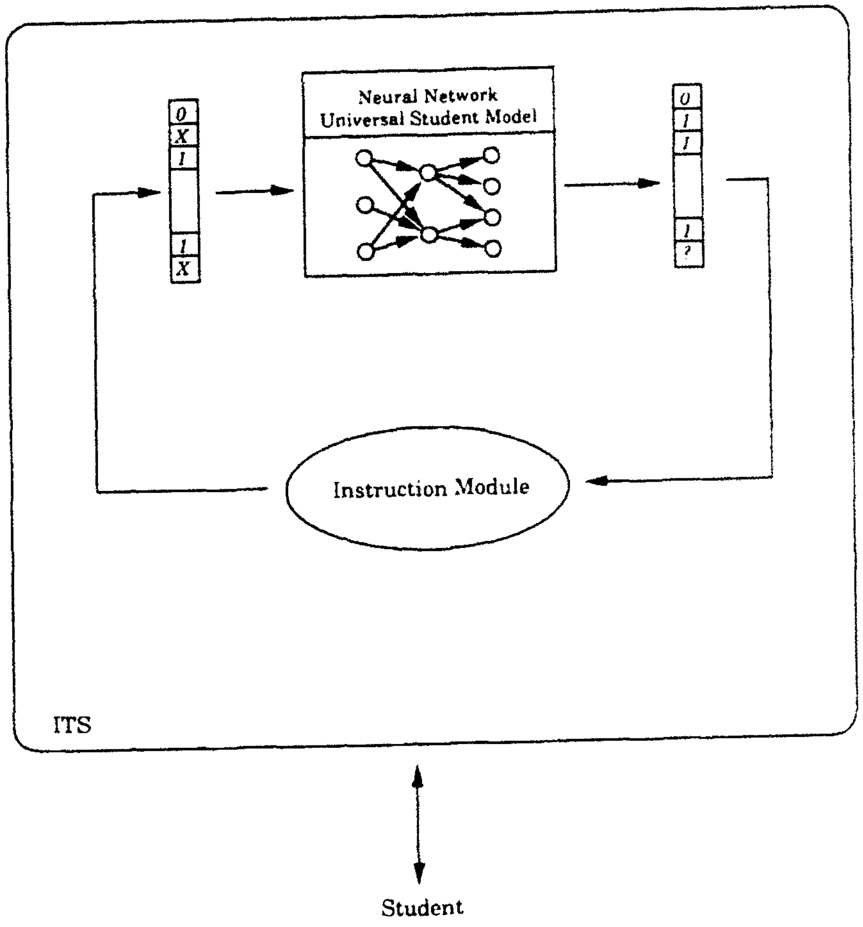

Figure 3
The neural network student module is utilized by the instruction module of the ITS.

STUDENT MODEL DEVELOPMENT WITH KOHONEN FEATURE MAPS

We have approached the problem of student modeling with a generalization of Kohonen's self-organizing feature map model (Kohonen, 1984). The feature map is an unsupervised learning network that preserves topological information about the input space. The map is conventionally configured with a geometrical arrangement of units—e.g., a square lattice—and an interconnectivity pattern that implements a "winner-take-all" network. Given an input stimulus, one unit (the "winner") is activated; the winner has the weight vector that is closest (in Euclidean space) to the input stimulus, which can be either a real-valued or binary vector. The degree of match m_k of a unit k to an input can be computed as the inner product $\mathbf{x} \bullet \mathbf{w}_k$ of the input stimulus vector $\mathbf{x}$ (problem responses in our application) and the unit's weight vector $\mathbf{w}_k$. More simply without requiring normalized weights and inputs, the squared Euclidean distance (an inverted measure) is often used:

$$m^k = \sum_{i=1}^{n} \left(x_i - w_i^k \right)^2 \qquad (1)$$

where n is the dimensionality of the input stimulus, x_i is the value of the ith input, and w_i^k is the weight between the map unit k and the ith input note. Note that a perfect match occurs when each component of a unit's weight vector is identical to the corresponding component of the input vector.

The training process iterates three steps: presenting an input stimulus, determining the winning unit, and updating feature map weights. For weight modification, we adopt the following commonly used formula (e.g., Ritter, Martinetz, and Schulten, 1989):

$$\Delta w_i^k(t) = \eta(t) e^{-\frac{d^2(k,c)}{2\sigma^2(t)}} \left(x_i(t) - w_i^k(t) \right)$$

where $d(k,c)$ is the distance between units k and the winning unit c on the map (for example, with a feature map organized as a two-dimensional grid, $d()$ is the Euclidean distance between the coordinates of the two units), and η and σ are parameters, the learning rate and neighborhood width respectively, that are geometrically reduced as a function of the training iteration t:

$$\eta(t) = \eta_i \left(\frac{\eta_f}{\eta_i} \right)^{\frac{t}{T}}$$

$$\sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i} \right)^{\frac{t}{T}}$$

where T is the prespecified number of iterations of the training process, and η_i and σ_i (η_f and σ_f) are initial (final) values for η and σ.

After training, the feature map is utilized as an on-line classifier. The matching process of Eq. (1) is used here too: the winning unit identifies a cluster or category to an input stimulus.

Previous work with Kohonen feature maps has required that complete input vectors be presented to the network—the input stimulus must specify a value for each dimension. This constraint is inconvenient for our application: we foresee the MUSK as a predictor of a student's knowledge state from an incomplete input. Given responses to some subset of the problems, we need to predict the student's knowledge state.

To accommodate missing information in input stimuli, we note that the matching operation in Kohonen feature maps involves an independent comparison in each input dimension. The match can therefore be performed in the subspace defined by the available problems:

$$m^k = \sum_{i \varepsilon P \subset K} \left(x_i - w_i^k\right)^2 \tag{2}$$

where P is the subset of the universal problem set K for which responses x_i are provided.

Partial data is useful for model development as well. In many testing situations, individual students are given some, possibly randomly-determined, subset of a universal problem set. (For example, the universal set could contain a number of different problems for each underlying concept.) The resulting problem response database is thus a set of vectors in each of which there are several elements that have no value. Due to the redundancy in the problem set, however, there is sufficient information in the data to develop an accurate student model. For feature map training with partial data, we use Eq. (2) for the matching step. Ties for the winner are broken randomly and weight modifications are limited to the P problem units. For more discussion on training with partial data, as well as some experimental results, see Samad and Harp (1992).

Each unit in the trained network represents a cluster of students: a region of possible input values reflecting statistical regularities in the training data. The representation is explicit in the weights themselves—the weight value between an input unit and a map unit is the prototypical value of the input for the cluster. (With input values constrained to the [0,1] internal, weight values will also be so constrained. As noted below, continuous-valued weights are thresholded for a binary interpretation.)

A map unit thus represents a knowledge state and the weights associated with the unit indicate whether or not a student in that state is capable of correctly answering the problems. Unlike, say, multilayer perceptron neural networks, Kohonen feature maps are not virtually uninterpretable "black boxes."

The perspicuity of representation is a useful feature for this application. It enables MUSKSs to realize some powerful capabilities:

- After training, once the best matching unit in the problem response subspace is determined, predictions for the remaining problems are easily produced.

If unit c has the closest match, and p_j is the prediction for problem j ($j \notin P$), then:

$$p_j = \begin{cases} 1 & \text{if } w_j^c \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

- Similarly, the weights indicate the implications of any knowledge state. This feature provides the model some noise immunity and error correcting ability. An imperfect match with the winning unit in the space of available problem responses may indicate careless errors and/or luck guesses.

- Knowledge representation in the feature map is localized—individual map units correspond to different knowledge states. The matching operation is conducted in parallel over all units. Degrees of match are available for all knowledge states and these car serve as "confidence factors" for predictions associated with particular knowledge states. Thus the predictions from Eq. (3) would be weighted by m^c.

A MUSK can be presented as a lattice of the knowledge states in a *Hasse diagram* (Trotter, 1983). The graph is completely determined by the subset relation between the knowledge states, a partial order, and is easily generated once the states have been identified. A node, N_X, in the MUSK represents a particula, knowledge state X. In effect, the graph must enforce one constraint: node N_A is an ancestor of N_B if, and only if, *state A is a superset of state B.* By convention, no arc is drawn between two nodes when a third, intermediate, state is properly contained in states of the larger of them. A simple algorithm generates the graph from the set of states.

Because the graph can be derived from the states, the central problem of our concern has been the identification of the clusters. Noise and incomplete data prevent immediate enumeration of the states. The ideal feature map topology should maximize the likelihood of capturing the correct knowledge states from such data. We currently adopt a standard feature map topology: a string. The string topology is a reasonable default since it imposes relatively few constraints—a string network contains the minimum number of edges which ensures connectedness.

Despite the near-universal reliance on regular networks, feature map training can be done with arbitrary topologies. Preliminary experiments using a topology that is isomorphic to the problem domain produced positive results under this assumption (Samad and Harp, 1992). In this scenario, the network learns an isomorphism, mapping the nodes of the network to the states of the knowledge space. In other words, the network classifies the questions, whose properties may be unknown, into the states of a known knowledge structure.

EXAMPLE

We discuss next the application of this student model to a particular domain. We have chosen a domain that is relatively simple and for which sufficient expertise exists for determining plausible knowledge states. The expert-synthesized states can then be used for generating training data, and for comparison with the MUSK developed from the data.

Figure 4 shows a knowledge state graph for fuel management performed by the navigator of an AC-130H aircraft. This position in the AC-130H (a variant of the C-130 transport known as a Gunship, and used in Special Operations) is responsible for inputs to the flight and fuel planning necessary to accomplish the mission. Aircrew members for this aircraft come from the ranks of experienced C-130 crews. Therefore, mission fuel planning in general is not a new task. What makes this particular problem unique is the kinds of mission events and conditions that must be planned for (e.g., combat loiter), the kinds of contingencies that must be anticipated, and unplanned changes in mission conditions that must be solved as part of the planning process. The knowledge states shown represent some of the competencies needed to prepare and manage the mission fuel plan for a tactical mission. There are six primitive concepts for the domain, listed in increasing complexity:

- C-5. The correction for air density applied to equivalent airspeed. This task is performed for a given temperature and pressure to determine true airspeed for time/distance/fuel calculations.

- C-4. Air speed calibration for cruise at 200 knots indicated airspeed. This task may be represented by the problem statement: For a given cruise altitude and cruise indicated airspeed, find the calibrated airspeed at the indicated airspeed.

- C-3. Cruise altitude determination for a given set of conditions. For a given gross aircraft weight, cruise power setting, temperature, and non-zero drag index, find the recommended or most efficient cruise altitude.

- C-2. Calculation of maximum endurance calibrated airspeed and fuel flow for a given "holding." This task may be represented by the problem statement: For a given pressure altitude, beginning holding weight, temperature, drag index, holding time, number of operational engines, and type of bleed, find the best average calibrated airspeed to be flown to minimize the amount of fuel used.

- C-1. Calculation of combat loiter time on station for a given amount of fuel. For a given pressure altitude, beginning loiter gross weight, aircraft configuration, fuel amount, and true airspeed, calculate the combat loiter time.

- C-0. Calculation of necessary fuel requirements for any or all segments of a given tactical mission. This task may be represented by the following problem statement: Given a flight plan for a tactical mission, including pressure altitudes, true airspeeds, take-off gross weight, temperature, drag index, winds, and other conditions that may be dictated by the mission, calculate the elapsed time and the amount of fuel needed for any or all flight segment.

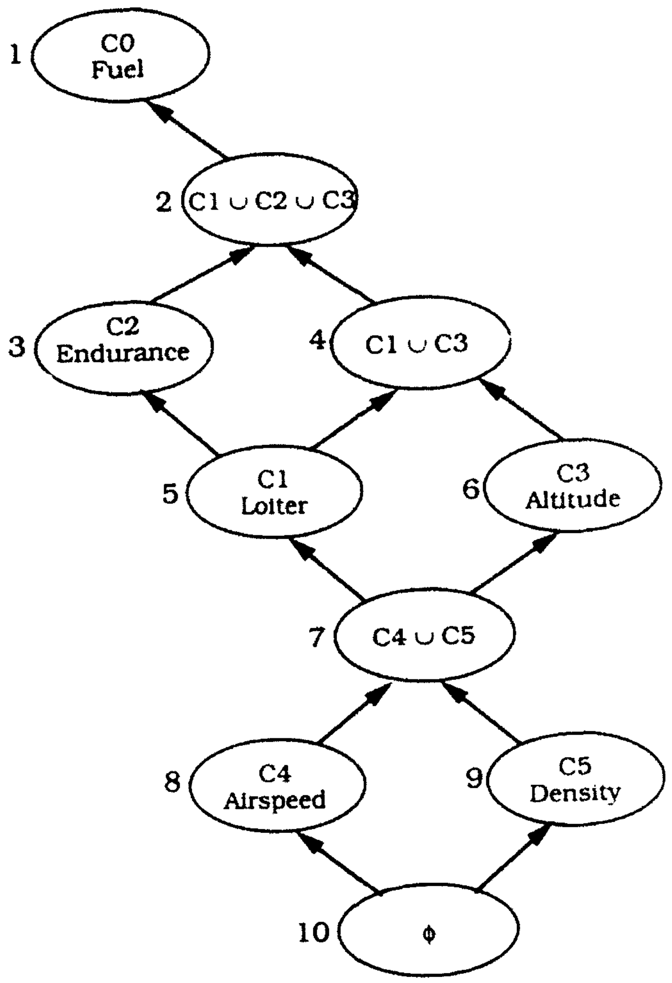

Figure 4
The correct MUSK model for fuel management performed by the navigator of an AC-130H aircraft.

As shown in Figure 4, we posit additional states which represent unions of these concepts. For example, state 7 represents a skill level in which the student is capable of both air density correction and air speed calibration (but not tasks involving higher-level concepts).

Based on Figure 4, a problem mastery matrix was generated (Table 1). For each knowledge state, the mastery matrix indicates which of the problems in the problem set a student in that state is capable of correctly answering. We assumed 25 problems in the set, and the entries are consistent with Figure 4. Thus state 9 represents complete ignorance, state 0 represents full understanding of the domain, and the mastery vector for a knowledge state is a superset of mastery vectors for all child states. In all our experiments, we assumed a population of 1000 students uniformly distributed over the 10 states. Training sets thus consisted of 1000 25-dimensional input vectors.

Table 1. Problem Mastery Matrix for AC-130H Fuel Management Knowledge States.

Knowledge State	Problem Responses																								
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1	0	1	1	1	1
2	0	1	1	1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	1	1	0	1	1	1	1
3	1	0	1	0	0	1	1	0	1	1	1	0	0	1	0	1	1	0	0	1	0	1	1	1	1
4	0	0	1	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	0	1	0	1	1	1	1
5	1	0	1	0	0	1	1	0	1	1	1	0	0	1	0	0	1	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1
7	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: Correct (1 entries) and incorrect (0 entries) responses to each of 25 problems for students in knowledge states 0 through 9 indicated.

EXPERIMENTAL RESULTS

In a large number of experiments, we have been able to reliably achieve correct clustering from problem response data generated from Table 1. Experiments have been conducted with varying noise levels, and various choices of learning algorithm parameters. With significant noise in the training data, the final values of network weights are not zero or one. In these cases, we threshold the weights at 0.5 to generate binary response vectors for all knowledge states. Some anecdotal evidence indicates that the proximity to 0.5 is a monotonic function of the noise probability for a problem; we have yet to systematically investigate this hypothesis.

We have been impressed by the robustness of the approach. It displays noise tolerance, and the learning parameter values are not critical. To assess robustness quantitatively, we have conducted a large number of experiments with varying noise and parameter values. In each experiment, a number of runs were performed with different random number generator seeds. The proportion of runs in each experiment that resulted in the correct (i.e., perfect) student model were computed. This is a severe criterion: an incorrect prediction of one response in one state disqualifies the trained feature map.

Noise in the context of problem response data consists of careless errors (incorrect answers where the mastery matrix would predict correct ones) and lucky guesses (the converse). Figure 5 shows modeling accuracy as a function of noise. The noise factor is the probability for both careless errors and lucky guesses on any given question. The same probability of error was applied to all problem responses. With up to 10% noise, there is generally a better than 90% chance of obtaining the correct model.

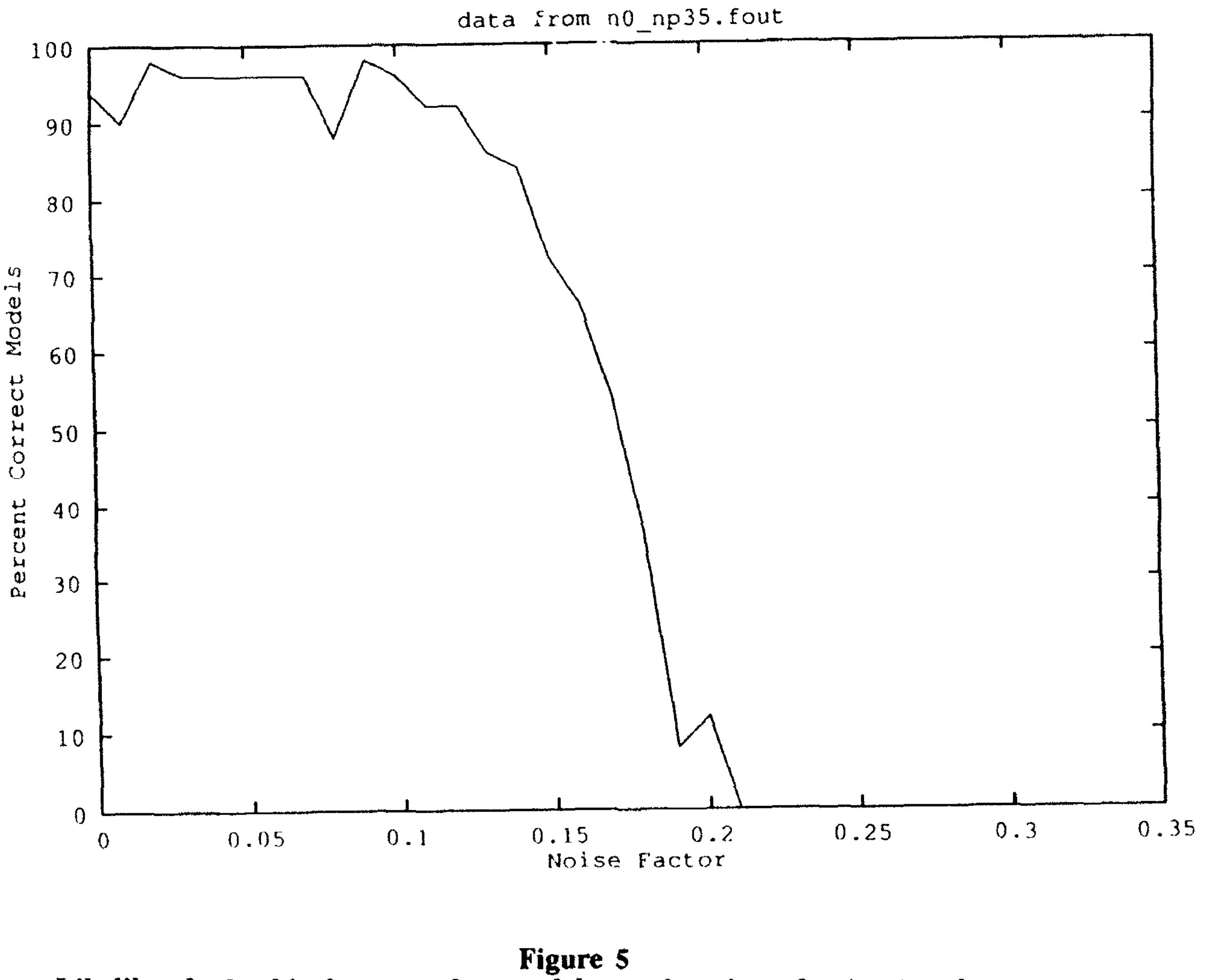

Figure 5
Likelihood of achieving a perfect model as a function of noise (careless errors and lucky guesses) in the response data. Learning algorithm parameters were as follows: $\eta_i = 0.2$, $\eta_f = 0.1$, $\sigma_i = 5$, $\sigma_f = 0.05$, $T = 10000$. Each point in the graph represents an average over 50 runs.

We observed an interesting, and somewhat paradoxical, phenomenon with very low learning rates. Figures 6a and 6b show performance as a function of noise for different parameter values. Performance on noise-free data is significantly poorer. Examination of the networks in these cases suggests a possible resolution. When incorrect models were produced, one unit of the feature map was not activated: it was not a winning unit for any training stimulus. Figure 7a schematically depicts the situation. One unit lies closest to two stimuli. With low learning rates and final neighborhood widths significantly less than unity, the unactivated unit will not be the closest match for either of the stimuli. Figure 7a thus represents a local minimum.

With noise present in the training data, however, the map of Figure 7a is less likely to be a stable final configuration. The variety of stimuli result in a greater likelihood of activating all units. The global minimum solution (Fig. 7b) can thereby be achieved. Note that the feature map of Figure 7b is stable for both the noise-free and the depicted noisy training spaces. The map of Figure 7a, however, is only stable for the noise-free data.

In general, the probabilities of careless errors and lucky guesses may differ. Depending on the construction of the test and student instructions, one or the other will be more likely. For example, lucky guesses are considerably more likely than careless errors in multiple-choice tests. For "fill-in-the-blank" problems, on the other hand, the converse is true. Figure 8 shows how model development is affected by independent careless error and lucky guess probabilities.

The approach is also reasonably insensitive to parameter selections. We expended little effort in attempting to determine optimal values for the learning algorithm parameters. Subsequent experiments confirmed that the ease with which we found appropriate values was a consequence of the parameter robustness of the model. Some of these experiments are summarized in Figures 9, 10, and 11. In each case two parameters are varied over a wide range—up to two orders of magnitude—and the frequency with which a perfect model was obtained is shown. Over a broad range of parameters, the correct model is realized with near certainty.

DETERMINING THE CORRECT SIZE FOR THE FEATURE MAP

As with many other neural network architectures, feature map application consists of more than network training. Before the learning algorithm can be executed, the network structure must be determined. The availability of an algorithm for generating the student model structure from identified clusters allows us to use a string topology for the map. There remains the problem of determining the size of the network: the number of feature map units that constitute the string. The synthetic nature of our application specifies a unique correct solution. In practice, the number of states underlying a problem response database is typically not well-defined. Expert assessments of this quantity can differ substantially. Kambouri et al. (1991) observed an order of magnitude variation in expert estimates of the number of states in a high-school mathematics domain. We have investigated two approaches to the problem of estimating the number of states.

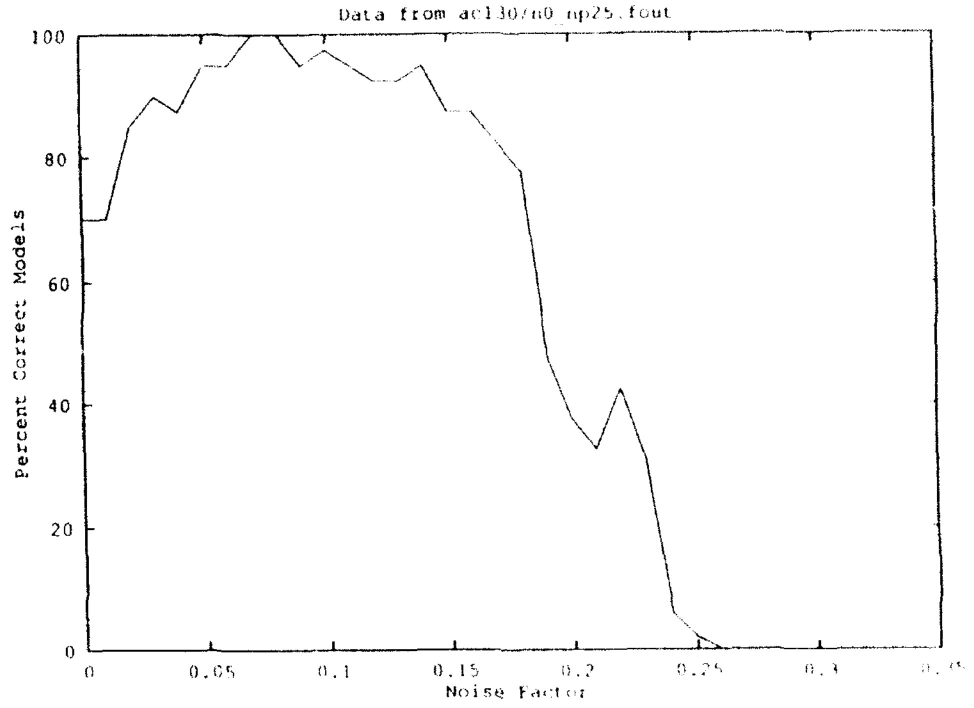

Figure 6a

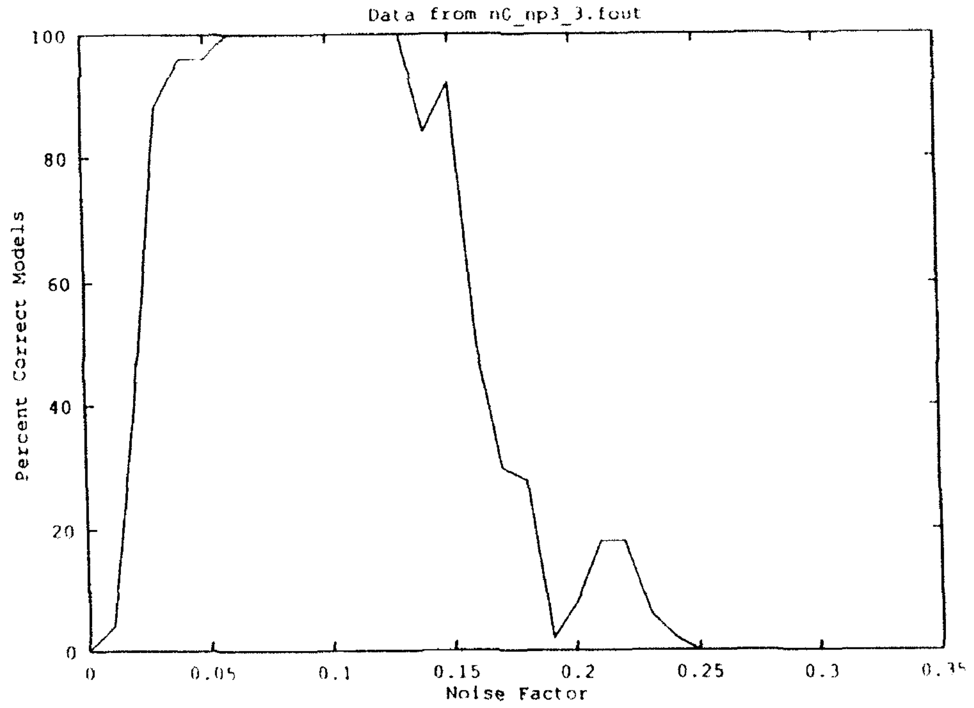

Figure 6b

Likelihood of achieving a perfect model is a non-monotonic function of noise in some parameter regimes. Learning algorithm parameters were as follows: For (a), η_i = 1.0, η_f = 0.01, σ_i = 10, σ_f = 0.1, T = 10000; For (b), η_i = 0.05, η_f = 0.01, σ_i = 10, σ_f = 0.1, T = 10000. Each point in graph (a) is an average over at least 20 runs; each point in graph (b) is an averge over 50 runs.

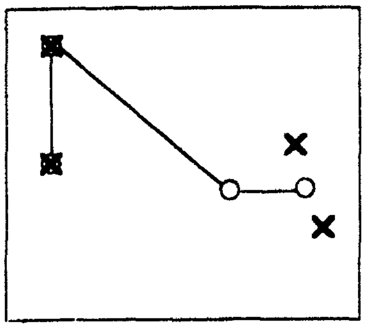

Figure 7a

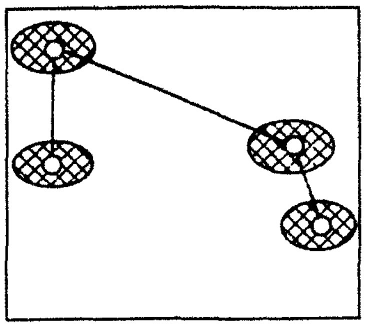

Figure 7b

Noise in training data for unsupervised learning can help avoid local minima. In (a), the "X" represent (noise-free) training *stimuli*. In (b), *noisy training* stimuli are generated from regions of the input space, not points. The trained map configuration in (a) is a local minimum for noise-free data, but is not a stable point with noisy data. The trained map configuration in (b) is a stable global minimum for both noise-free and noisy data.

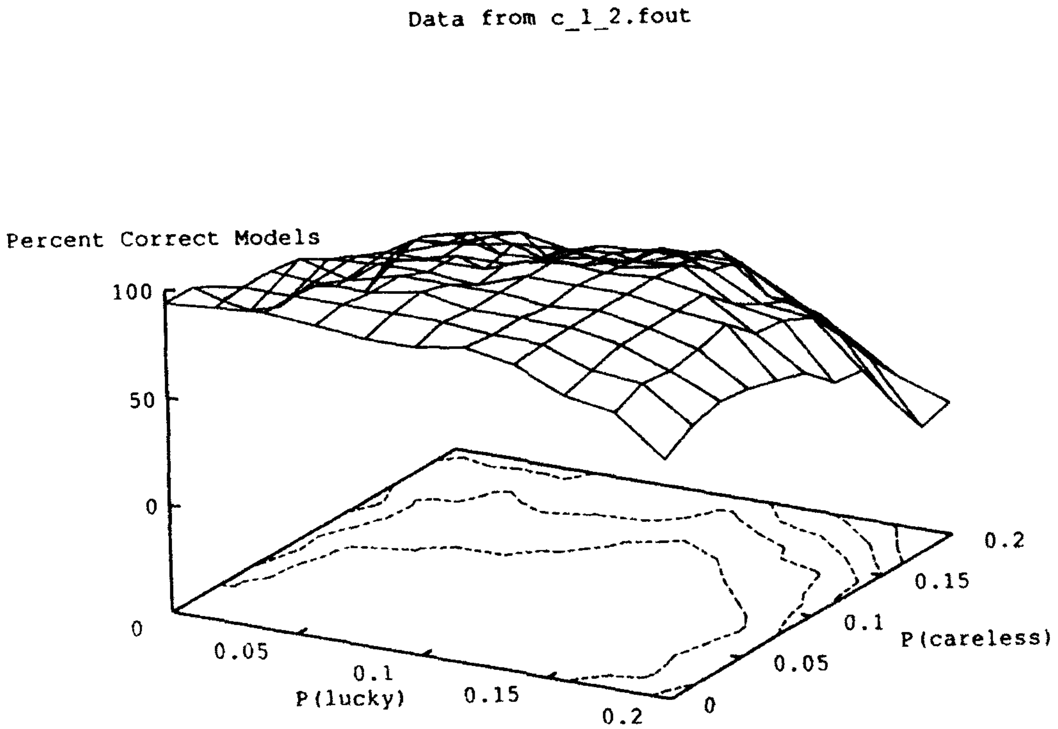

Figure 8
Likelihood of achieving the perfect model as a function of two independent noise parameters: probabilities for careless errors and lucky guesses. Learning algorithm parameters were as follows: $\eta_i = 0.2$, $\eta_f = 0.1$, $\sigma_i = 5$, $\sigma_f = 0.05$, $T = 10000$. Each point in the graph represents an average over 50 runs. Contour lines are at 23.3%, 38.6%, 54%, 68.3%, and 84.6%.

Data from ei_ef_4.fout

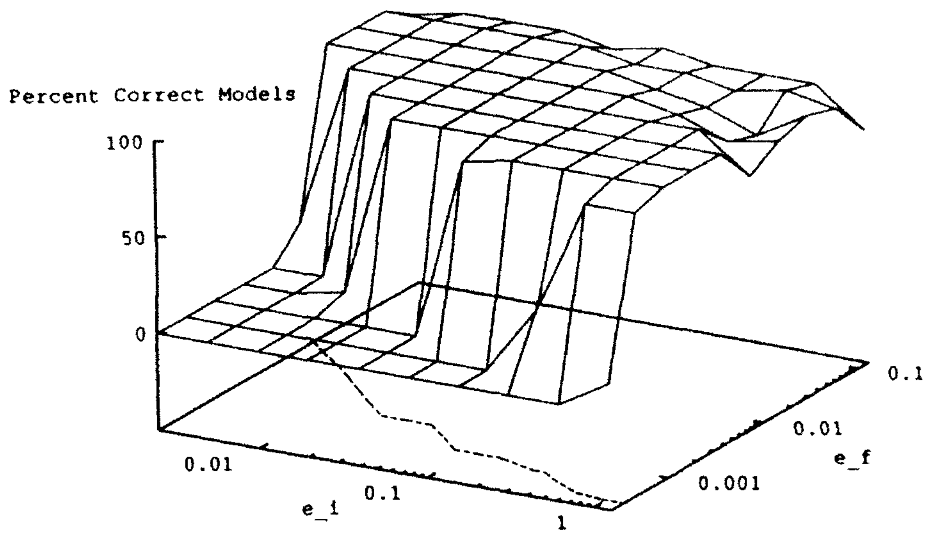

Figure 9
Likelihood of achieving the perfect model as a function of initial (e_i or η_i) and final (e_f or η_f) learning rate parameters. Other parameters were as follows: $\sigma_i = 10$, $\sigma_f = 0.01$, T = 10000, 10% noise. Each point in the graph represents an average over 20 runs. Contour lines are at 16.7%, 33.4%, 50%, 66.7%, and 83.4%.

Data from ei_ef_5.fout

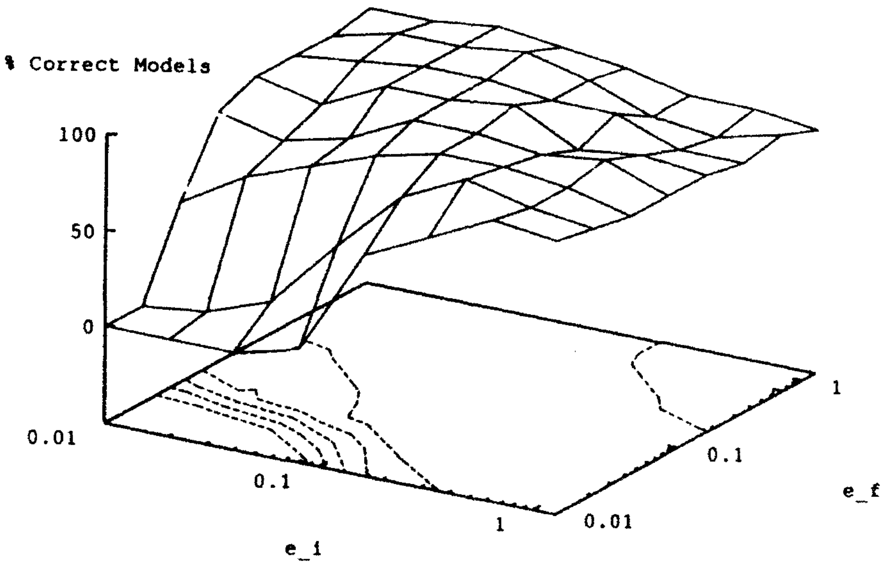

Figure 10
Likelihood of achieving the perfect model as a function of initial (e_i for η_i) and final (e_f or η_f) learning rate parameters. Other parameters were as follows: $\sigma_i = 5$, $\sigma_f = 0.05$, T = 10000, no noise. Each point in the graph represents an average over 25 runs. Contour lines are at 16.7%, 33.4%, 50%, 66.7%, and 83.4%.

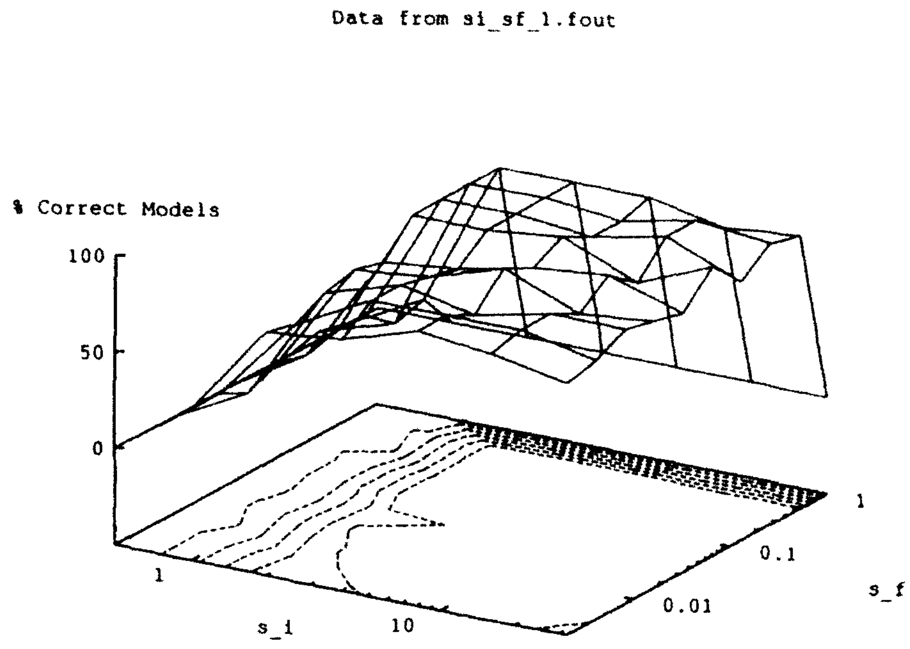

Figure 11
Likelihood of achieving the perfect model as a function of initial (s_i or σ_i) and final (s_f or σ_f) neighborhood parameters. Other parameters were as follows: $\eta_i = 0.2$, $\eta_f = 0.1$, T = 10000, no noise. Each point in the graph represents an average over 25 runs. Contour lines are at 16.7%, 33.4%, 50%, 66.7%, and 83.4%.

An obvious measure of clustering accuracy is the rms error e between stimuli and weight vectors of corresponding winning map units which by itself is inadequate: it is typically monotonic with increasing units. In an approach similar to that used for fitting data smoothing models, we penalize this measure with a complexity term, the number of map units m. The combined minimization criterion that is used to determine the optimal number of map units n^* is:

$$n^* = \arg_n \min J(e,n)$$

$$J(e,n) = a_1 e + a_2 n$$

The a_i are adjustable coefficients. With a penalty on larger networks, this approach is effective. Figure 12 shows the result of evaluating the cost function J over a range of n. The minimum is obtained for $n=10$. When n is small, the result is not critically contingent on values of the coefficients. The correct minimum is obtained for all a_1, a_2 such that $26 \leq a_1/a_2 \leq 250$, almost an order of magnitude range in ratios. For larger n, another constraint applies: the weight vectors of the learned network should all be unique. In our case, for $n=13$, three pairs of duplicate weight vectors were observed. The ten distinct weight vectors were the correct representations for the ten knowledge states. The search for optimal network size need not be exhaustive. Efficiency is important for knowledge spaces with large numbers of states. We have used a line search algorithm (golden section) to rapidly converge on appropriate topologies.

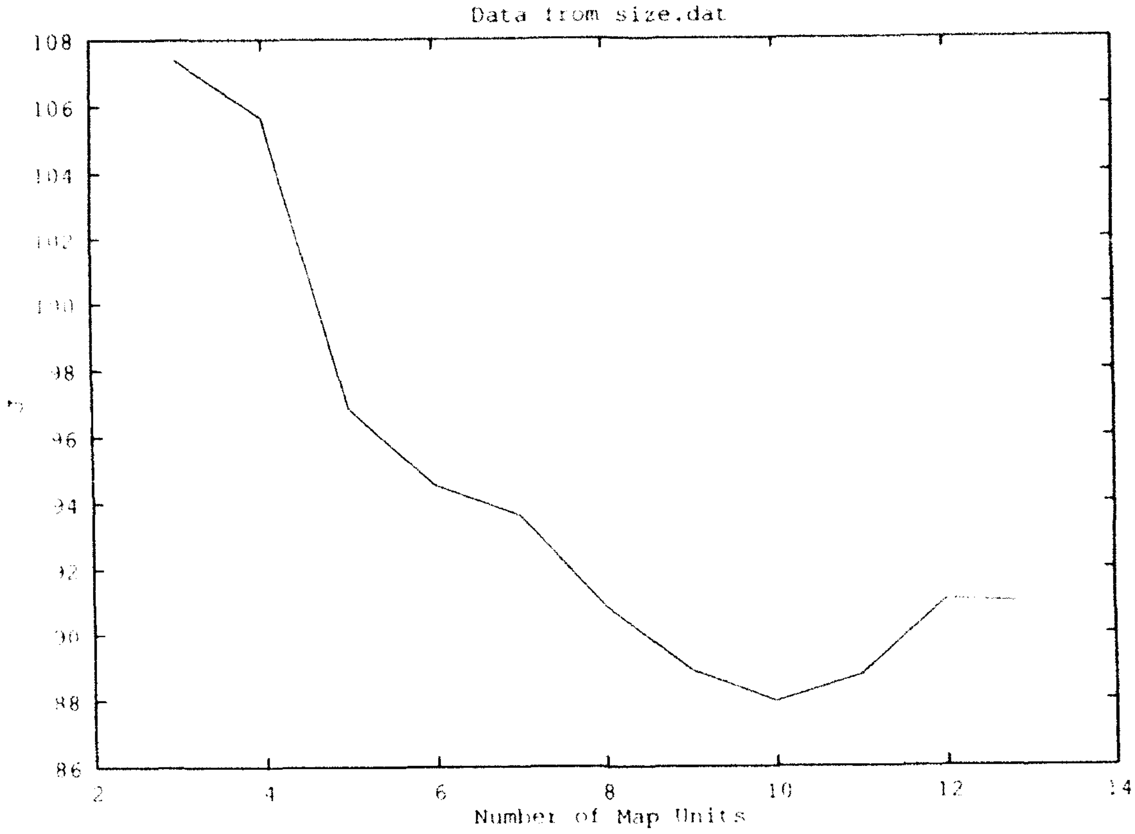

Figure 12
Values of the cost function *J* as a function of network size. The minimum of *J* is encountered for the correct number of feature map units: 10. Parameters as follows: $\eta_i = 0.05$, $\eta_f = 0.01$, $\sigma_i = 10$,$\sigma_f = 0.1$, T = 10000, noise factor = 10%.

This approach performs reliably for small to moderate domains (less than 50 states). As the number of states grows larger, however, local minima associated with the penalty function *J* can confound the simple line search technique. This problem is illustrated in a randomly generated knowledge space of 50 states. The example presumed 10 underlying items, and two questions on each, so each map unit was of dimension 20. Feature maps of varying lengths were trained on a randomly generated database of 1000 students; student responses included a "lucky guess" rate of 0.2. Networks were given 10,000 training trials. With coefficients $a_1 = 0.99$ and $a_2 = 0.01$, the line search algorithm sometimes settles on a near optimal string length of 51 (see Figure 13). However, the function J offers an even lower point at length 70, which is also a plausible (but incorrect) solution.

An alternative approach follows from considering what happens when the size of the network is much greater than the number of states in the underlying knowledge structure. After training, a large number of units are never "winners"; the remaining units, each of which is optimal for at least one student, provide a good estimate of the number of states in the knowledge structure. Indeed, these winning units can be isolated to recover the elements of the knowledge structure. Thus, to simultaneously estimate the size of the space and extract its elements, it suffices to incrementally increase the network size until further increases cease to increase the number of activated units. In the example of Figure 13, the asymptote of the upper curve is precisely 50. This approach has successfully been applied to randomly generated knowledge structures of sizes 50, 100, 200, and 400 elements.

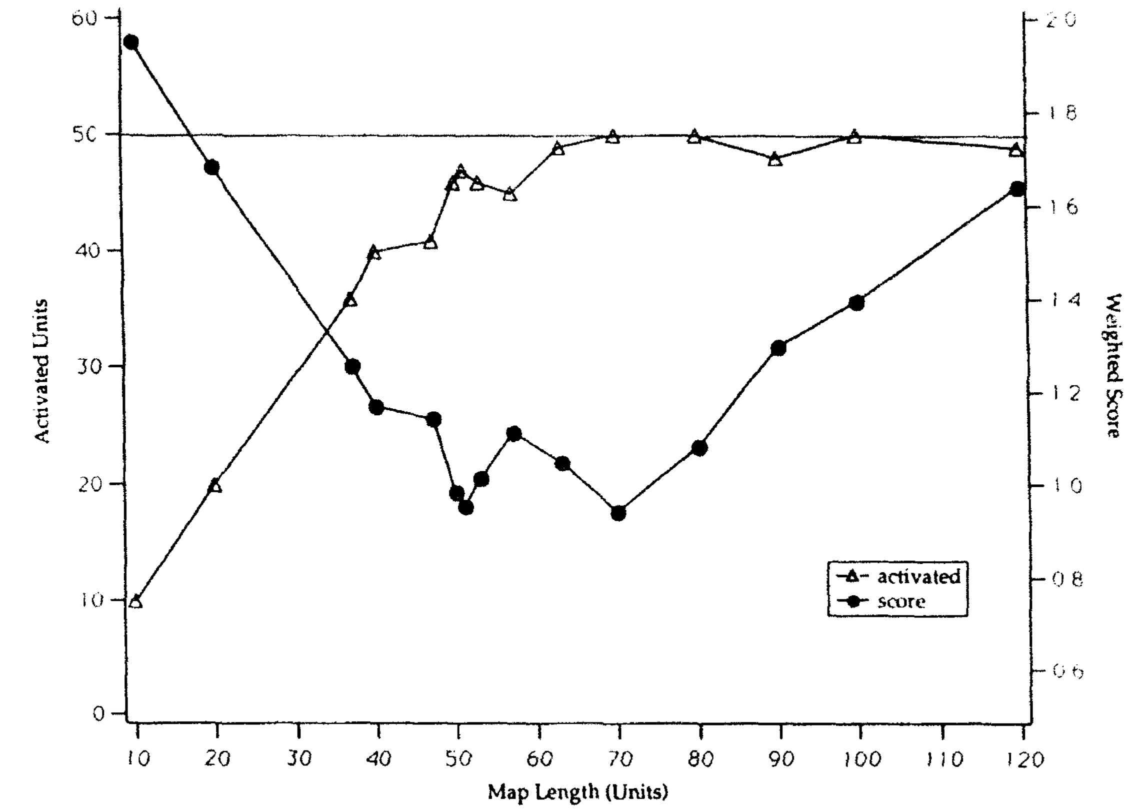

Figure 13
Penalized error function (right axis) and activated units (left axis) as a function of the number of the map units. The networks were trained on a synthetic knowledge structure of 50 randomly generated states. Noise factor (lucky guess rate) = 20%. Other parameters: $\eta_i = 0.2$, $\eta_f = 0.1$, $\sigma_i = 5$, $\sigma_f = 0.05$, T = 10000.

APPLICATIONS OF MUSK

As noted earlier, our approach to student modeling can be placed within the framework of Knowledge Space Theory (KST) and thus, ultimately imbedded in an Intelligent Tutoring System (Villano, 1992; Villano & Bloom, 1992). The student model in an ITS provides support for the following functions: adaptively assessing the student's mastery of the material, charting the student's progress through the curriculum, selecting the appropriate level of hinting and explanation, and facilitating student feedback. In addition to dynamically adapting to the student's responses during an interaction with the ITS, the student model should also utilize prior assessment data obtained from a population of students as illustrated by the MUSK approach.

Several factors contribute to uncertainty in student modeling such as careless errors and lucky guesses in the student's responses, changes in the student knowledge due to learning and forgetting, and patterns of student responses unanticipated by the designer of the student model. The motivation for a *probabilistic* student model stems from the need to represent this uncertainty regarding the estimate of the student's knowledge. A promising approach to the development of a probabilistic student model is to incorporate probability measures in the MUSK approach.

In KST, stochastic knowledge assessment routines have been developed in which uncertainty regarding the student's knowledge state is represented by a probability distribution on the states (Villano et al., 1989; Falmagne & Doignon, 1988; Villano, 1991). The assessment routine updates the probability distribution on the states to be consistent with the student's responses to a carefully chosen sequence of problems. The probability of a correct response to a problem can also be computed from the distribution on the states. In addition, lucky guess and careless error probabilities can also be estimated easily, given the MUSK model and the problem response database. Below, we briefly summarize some potential applications of our probabilistic student model concept within an Intelligent Tutoring System.

Problem Selection. For an ITS to be adaptive, it must be capable of determining the next "best" problem to pose to the student based on a dynamic student model. In KST, one method for selecting the most "informative" problem is available that can straightforwardly be applied to MUSK models: choose the problem with the least predictable response (Falmagne & Doignon, 1988; Villano, 1991). For the *half-split* selection rule, we choose the problem whose probability of being answered correctly, $\rho(q)$ is closest to .5. The reasoning is as follows. If $\rho(a) = .85$, then problem *a* would not be very informative because it is almost certain the student would respond correctly. If $\rho(d) = .1$, then problem *d* would *not* be informative because we are fairly certain $(1-.1 = .9)$ that the student would fail this problem. If $\rho(c) = .5$, then problem *c* would be the most informative problem to ask of these three because there would be an equal chance of the student passing or failing problem *c*. Problem *c* is thus the problem for which our estimate of how the student will respond is the most uncertain.

Curriculum and Advancement. A MUSK model represents various learning paths through a curriculum, and can thereby accommodate different instructional strategies of educators and different learning styles on the part of students. The learning paths (gradations) may be used to guide the progression of the student through the curriculum. In a well-graded knowledge space, the next lesson to teach is the one tested by the next problem in the learning path. In the event that there is more than one path to follow from the current knowledge state, you may choose the path to the easiest problem (the problem with the highest probability of being answered correctly), or the problem along the most traveled (or most probable) learning path. The student would be expected to master the current problem in the learning path before moving on to the next problem. Mastery of a "problem" could be defined by a score on equivalence class of test problems or task.

Hint Level. The content and nature of the hint or explanation in an ITS relies upon the student model's representation of the student's level of mastery of the material. More advanced students may be given terse explanations, whereas novice students could be given more elaborate guidance. The level of mastery is readily available in a

MUSK model. A measure of problem difficulty that can be useful in this context is the "height" of a problem as used in KST. The ***height*** can be defined by the minimal number of problems which must be solved before a particular problem, and can be used to determine the level of hinting. A problem parameter such as the probability of a careless error may also influence hinting. For example, if a problem had a relatively high probability of a careless error, a hint might warn the student to take extra time to check and confirm the answer to the problem.

Student Feedback. An inspectable, detailed representation of the learner's mastery of the material can provide feedback regarding the student's most recent accomplishments and most pressing weaknesses. With a MUSK model, rather than reporting a single score (i.e., ability = 95%), we can be much more specific and indicate the most advanced problem that has been mastered as well as a list of the missed problems and/or future problems to be mastered. We would prefer not to lose the distinction between a student who can answer many simple problems versus one who can answer a few difficult problems. A graphical representation of the knowledge structure may also be used as an inspectable student model indicating the student's position in the curriculum.

CONCLUSIONS

The MUSK approach to student model generation provides a novel approach to take advantage of existing databases. Raw data can be processed by the self organizing feature map, yielding a characterization of the knowledge structure, which can be used to classify students in the ITS. With the synthetic problem examined, it has proven to be robust and easy to apply. There are open issues as to how the technique will perform with raw data representing very large knowledge structures, and how best to interface a MUSK to an ITS. To explore the former question, we are currently examining a real test database from the New York Regents examination in mathematics containing results from 60,000 students.

ACKNOWLEDGMENTS

We gratefully acknowledge the assistance of T. Sadani in the design and development of our simulation software and P. K. Watson of Continental Systems Technology in developing the aircraft domain. C. Bloom contributed to our understanding of ITS applications of MUSKs.

REFERENCES

Doignon, J.-P., & Falmagne, J.-C. (1985). Spaces for the assessment of knowledge. *International Journal of Man-Machine Studies, 23*, 175-196.

Falmagne, J.-C. (1989). A latent trait theory via a stochastic learning theory for a knowledge space. *Psychometrika, 54*, 283-303.

Falmagne, J.-C., Doignon, J.-P., Koppen, M., Villano, M., & Johannesen, L. (1990). Introduction to knowledge spaces: How to build, test and search them. *Psychological Review, 97*, 201-224.

Falmagne, J.-C., & Doignon, J.-P. (1988). A class of stochastic procedures for the assessment of knowledge. *British Journal of Mathematical and Statistical Psychology, 41*, 1-23.

Kambouri, M., Koppen, M., Villano, M., & Falmagne, J.-C. (1991). *Knowledge assessment: Tapping human expertise by the QUERY routine.* Tech. Rep. No. MBS 91-20, Irvine: University of California, Irvine Research Unit in Mathematical Behavioral Sciences.

Kohonen, T.E. (1984). *Self-Organization and Associative Memory.* New York: Springer-Verlag.

Lord, F.M. (1980). *Applications of item response theory to practical testing problems.* Hillside, NJ: Lawrence Erlbaum Associates.

Ritter, H.J., Martinez, T.M., & Schulten, K.J. (1989). Topology-conserving maps for learning visuo-motor-coordination. *Neural Networks, 2*, pp. 159-168.

Samad, T., & Harp, S.A. (In press). Self-organization with partial data. *Network: Computation in Neural Systems, 3*, May.

Sleeman, D., & Brown, J.S. (1982). *Intelligent Tutoring Systems.* New York: Harcourt, Brace, Jovanovich.

Trotter, W.T. (1983). Graphs and partially ordered sets. *In Graph Theory, 2*, L. Beineke, R. Wilson editors. London: Academic Press.

Villano, M. (In press). Probabilistic student models: Bayesian belief networks and knowledge space theory. Proceedings of the Second International Conference on Intelligent Tutoring Systems. New York: Springer-Verlag, *Lecture Notes in Computer Science*.

Villano, M. (1991). Computerized Knowledge Assessment: Building the knowledge structure and calibrating the assessment routine. (Doctoral dissertation, New York University, New York, 1991). *Dissertation Abstracts International*, 52-12B.

Villano, M., & Bloom, C. (In press). *Probabilistic Student Modeling with Knowledge Space Theory.*

Villano, M., Falmagne, J.-C., Johannesen, L., & Doignon, J.-P. (1987). Stochastic procedures for assessing an individual's state of knowledge. *In Proceedings of the International Conference on Computer Assisted Learning in Post-Secondary Education* (pp. 369-371). Calgary, Alberta, Canada: University of Calgary.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems.* Los Altos, CA: Morgan Kaufmann.

Weiss, D.J. (Ed.) (1983). *New Horizons in testing: Latent trait theory and computerized testing.* New York: Academic Press.

Wood, P.H., & Holt, P.D. (1990). *Intelligent tutoring systems: An annotated bibliography. ACM SIGART Bulletin, 1*, 1, 21-42.

•U S GOVERNMENT PRINTING OFFICE 1993-0-761-052/60071

PREFACE

The mission of the Intelligent Training Branch, Technical Training Research Division, Human Resources Directorate, Armstrong Laboratory (AL/HRTI) is to design, develop, and evaluate the application of artificial intelligence (AI) technologies to computer-assisted training systems. The current effort was undertaken as part of HRTI's research on intelligent tutoring systems (ITS) and ITS development tools. The work was accomplished under work unit 1121-09-81, Application of Artificial Neural Networks to Modeling Student Performance. The proposal for this research was solicited using a Broad Agency Announcement.